**<https://www.e-olymp.com/uk/problems/2987>**

**XV Всеукраїнська олімпіада з інформатики**

Багатокутники

На площині задана така множина з *N* багатокутників, що виконуються наступні умови:

1. ніякі два багатокутника не мають спільних точок;
2. для кожного *i*–го багатокутника існує *Pi* багатокутників, всередині яких він знаходиться, і   
   *N*-1-*Pi* багатокутників, котрі знаходяться всередині нього, 0≤*Pi*≤*N*-1.

Завдання

Напишіть програму POLYGON, яка для кожного багатокутника видає кількість багатокутників, всередині яких він знаходиться.

Вхідні дані

Перший рядок вхідного файлу POLYGON.DAT містить ціле число *N* — кількість багатокутників, 3≤*N*≤10000. Наступні *N* рядків файлу описують *N* багатокутників. (*i*+1)–ий рядок файлу описує *i*–ий багатокутник. Перше ціле число *Ci* — кількість вершин багатокутника, 3≤*Ci*≤20. Наступні *Ci* пар чисел — координати вершин багатокутника у порядку його обходу. Координати вершин — цілі числа, що належать діапазону від -2 000 000 000 до 2 000 000 000.

Вихідні дані

Єдиний рядок вихідного файлу POLYGON.SOL повинен містити N чисел: *i*–те число рядка повинно бути *Pi* — кількість багатокутників, всередині яких знаходиться *i*–ий багатокутник.

Приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| POLYGON.DAT | POLYGON.SOL |
| 3  3 -2 1 8 9 12 1  3 7 5 6 3 7 4  4 4 3 7 7 9 3 1 2 | 0 2 1 |

**Реализация:**  
Многоугольники не имеют общих точек. Очень важное условие.  
Заведем массив, в который для каждого многоугольника запишем его крайнюю правую координату (можно записать любую другую "самую", но мы разберем случай с правой координатой). Это самая правая точка многоугольника. Естественно, что у самого большого многоугольника самая правая точка будет лежать правее самых правых точек всех остальных многоугольников. Если бы это было не так, то многоугольники имели бы точки пересечения, что противоречит условию.  
Техническая реализация: Следует завести два массива, в один из которых записать координаты самых правых точек многоугольника (только координату по X, ту что отвечает за "правость"), а в другой - номера многоугольников. Затем отсортируем по убыванию массив правых координат с помощью быстрой сортировки (она есть в примерах А.Н. Никитина) вместе с элементами массива координат перемещая элементы массива номеров многоугольников. После того как массив отсортирован, заполним его элементы с правой координатой значениями, соответствующими номеру этого элемента - 1 (т.е. 0, 1, 2...). Теперь в этом массиве хранится число многоугольников, которые окружают данный многоугольник. Теперь с помощью другой процедуры быстрой сортировки отсортируем массив номеров по возрастанию, перемещая соответствующие элементы "правого" массива. Это делается для ускорения вывода, чтобы не искать каждый раз элемент в массиве заново.  
Теперь просто идя по "правому" массиву от 1 до количества многоугольников выводим значения - это и есть количество окружающих многоугольников.

Важным условием есть то что многоугольники не имеют общих точек.  
  
   1. Создадим массив размером 3хN;  
   2. В первый столбец занесём самую правую точку каждого многоугольника(можно "самую левую" или другую "самую" точку).  
   3. В второй столбец занесём номер многоугольника, так как на важна последовательность.  
   4. Отсортируем массив по спаданию самой правой точки.  
   5. Так как многоугольник точка которого лежит дальше всех не будет находиться в любом другом многоугольнику то в третий столбец заносим "0". Следующий многоугольник лежит только в одном другом многоугольнику заносим ему в третий столбец "1" и так до конца.  
   6. Так как нам нужно вывести в том же порядке в котором ми и получали данные то сортируем массив по втором столбу в который мы заносили номер многоугольник.  
   7. Выводим

#include<iostream>  #include<math.h> #include<vector> #include<algorithm> using namespace std;

int main() {    long long a,b,c,n,mas[10011][30],mas2[10011][3];    cin>>n;    for(int i=0;i<n;i++){        cin>>mas[i][0];        for(int j=1;j<=(mas[i][0]\*2);j++)        cin>>mas[i][j];    }  
**// Пункт №2**  
    for(int i=0;i<=n;i++){        mas2[i][0]=mas[i][1];    for(int j=1;j<=mas[i][0]\*2;j+=2){        if(mas2[i][0]<mas[i][j])        mas2[i][0]=mas[i][j];        }    }

***// Пункт №3***    for(int i=0;i<n;i++){mas2[i][1]=i+1;}

**// Пункт №4** for(int i=0;i<n;i++)    for(int j=0;j<n-1;j++){        if(mas2[j][0]<mas2[j+1][0]){            int temp;            temp=mas2[j][0];            mas2[j][0]=mas2[j+1][0];            mas2[j+1][0]=temp;                        temp=mas2[j][1];            mas2[j][1]=mas2[j+1][1];            mas2[j+1][1]=temp;                        temp=mas2[j][2];            mas2[j][2]=mas2[j+1][2];            mas2[j+1][2]=temp;        }         }  **// Пункт №5**  
    for(int i=0;i<=n;i++){mas2[i][2]=i;}

**// Пункт №6**  
     for(int i=0;i<n;i++)    for(int j=0;j<n-1;j++){        if(mas2[j][1]>mas2[j+1][1]){            int temp;            temp=mas2[j][0];            mas2[j][0]=mas2[j+1][0];            mas2[j+1][0]=temp;                        temp=mas2[j][1];            mas2[j][1]=mas2[j+1][1];            mas2[j+1][1]=temp;                        temp=mas2[j][2];            mas2[j][2]=mas2[j+1][2];            mas2[j+1][2]=temp;         }            }     **// Пункт №7**     for(int i=0;i<n-1;i++)    cout<<mas2[i][2]<<" ";    cout<<mas2[n-1][2]<<endl;    }